Sugandha Lohar, Jane Cleland-Huang, Alexandar Rasin, Patrick Mäder:

Live Study Proposal: Collecting Natural Language Trace Queries

Zuerst erschienen in:
REFSQ-JP 2015, REFSQ workshops, research method track, and poster track : joint proceedings of REFSQ-2015 workshops, research method track, and poster track : co-located with the 21st International Conference on Requirements Engineering: Foundation for Software Quality (REFSQ 2015) : Essen, Germany, March 23, 2015
URN: urn:nbn:de:0074-1342-0
Page 207-210

Live Study Proposal: Collecting Natural Language Trace Queries

Sugandha Lohar[1], Jane Cleland-Huang[1], Alexandar Rasin[1], Patrick Mäder[2]

[1]DePaul University, Chicago, IL 60422, USA;
[2]Ilmenau University of Technology, Ilmenau, Germany
sonul.123@gmail.com; jhuang@cs.depaul.edu

Abstract. **[Context & motivation:]** Software traceability links which are created during the development process are subsequently underutilized because project stakeholders lack the skills they need to compose trace queries. TiQi addresses this problem by accepting natural language trace queries and transforming them to executable SQL. **[Question/problem:]** The TiQi engine depends on the presence of a domain model. This can best be constructed through collecting samples from potential users. We are interested to learn what trace queries would be of interest to potential stakeholders and what terminology they would choose to express those queries. **[Principal ideas/results:]**In this live study we will demonstrate TiQi in action, and lead the participants through a series of carefully crafted 'what-if' scenarios designed to capture a variety of sample queries. **[Contribution:]** The study is expected to lead to a more extensive domain model which will improve the accuracy of TiQi's query transformation process.

1 Research Problem

In safety-critical domains traceability is prescribed by regulators [3] who typically require traceability paths to be established between hazards, faults, mitigating requirements, design, code, and test cases [5] etc. Manufacturers must invest significant effort in creating trace links; [6] however these links are often used purely for compliance purposes and are otherwise underutilized [3]. The potential benefits of using trace links to support activities such as impact analysis and regression test case selection go unrealized. One reason is that trace queries can be quite complex and cut across many different artifacts. For example, a typical trace query could synthesize data about faults, mitigating requirements, code, test cases, and test logs, as well as the trace links that connect them. This leads to significant complexity, especially as many software developers have only rudimentary skills at constructing structured queries using technologies such as SQL or XQuery [1, 2].

In our previous paper presented at the International Requirements Engineering Conference, entitled "TiQi: Towards Natural Language Trace Queries" [4], we presented a solution for transforming NL trace queries into SQL. Prior to engaging in this project we assumed that we could find and adopt an open source

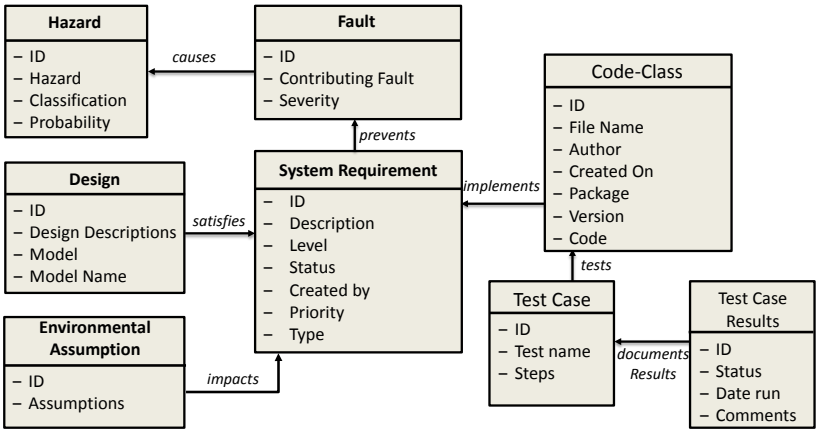

Fig. 1. A Traceability Information Model (TIM), depicting available artifacts and their associated traceability paths for the Isolette System.

general NL database query language and customize it for the traceability domain; however, an extensive search of available open-source solutions was unsuccessful. We therefore constructed a generic NL database query mechanism and then customized it with vocabulary and concepts from the traceability domain. The result of this work is *TiQi*, which transforms spoken or written natural language trace queries into executable SQL statements. In a series of controlled experiments, TiQi has transformed NL queries from different projects at accuracy rates ranging from 50% to 89%.

TiQi prompts the user for a NL query by displaying a Traceability Information Model (TIM), which as depicted in Figure 1 models artifact types, their attributes, and semantically typed links.

In order to interpret queries, TiQi requires an underlying domain model describing the terminology and concepts of the traceability and product domain. This includes *project-specific* terms, *question* terms and *junk*. *Project-specific* terms describe artifact types and attribute names, and can be extracted from the raw data of the project. *Question* terms form the 'glue' which holds the pieces of the query together. For example terms such as *show me*, *list all*, or *I'd like to see* are all synonyms for the SQL construct *SELECT*. Similarly, terms such as *associated with*, *related to*, or *with* are synonyms for various forms of *JOIN*.

Our **research goal** is to collect sample trace queries from which we can extend TiQi's domain model. There are two specific **research questions**. First, "What queries are of interest to project stakeholders?" and second "What terminology do project stakeholders use to express their queries?"

2 Research Design

The study will be performed interactively as a large group with both a paper-based and online option. The session will open with 10 minutes of training in which the notion of trace queries and TIMs will be presented. First, demographic data about each participant will be collected. Then each participant will be given a TIM similar to the one shown in Figure 1. Half of the participants will work on the Isolette System and half on the EasyClinic system [4]. Three data records will be provided as samples for each artifact type.

In the first part of the study specific scenarios will be used as prompts. The participant will be asked to address the scenario by creating a NL query – using words of their own choosing. There are nine scenarios of which four are shown below. Each participant will be asked to select and create a query for five of them. Participants will also be asked to add 1-2 of their own queries that would support a Software Engineering task which they perform regularly. Sample scenarios include:

1. The **safety officer** is worried that an important requirement $R136$ is not correctly implemented. The developer tells him that it is not only implemented but has also passed its acceptance tests. The security officer runs a trace query to confirm this. What might his query be?
2. **Piotr** discovered that one of the high risk hazards have been neglected during system specification. He cannot find any related requirements. Now he is worried that other hazards might have been neglected too. Can you help him issue a trace query to overcome his concern?
3. Last week, **Jin** modified some functions in order to improve the efficiency of the system. The system has not functioned well since then. She is not sure if all the test cases have passed. What trace query should she issue in order to verify this?
4. While testing the system, **Paxton** becomes suspicious that the test cases aren't sufficient to show that all requirements have been addressed. What trace query could he issue to help him investigate this problem?

To add some fun to the session – once all queries have been collected, we will randomly select three of them and feed them to TiQi. The participants will experience TiQi's capabilities and/or failures in real time. In terms of **benefits**, all participants will learn more about natural language trace queries and will be given access to our online TiQi tool (not yet released publicly).

Because our goal is to capture sample queries from a broad range of potential project stakeholders, we have only **basic prerequisites**. We require all participants to hold an MS degree in Computer Science, Information Systems, Software Engineering or a related field, and/or to have at least one year of IT experience. We will capture this information during the session through the meta-questions and will later discard any responses from people who do not meet these criteria.

To **analyze the data** we will use data mining tools to generate a list of terms, phrases, and their frequencies. We will then automatically filter the list

to exclude project specific terms and to remove terms that are already known to TiQi. This will leave the new question terms and the non-useful junk terms. Researchers on our team will then go through this list and categorize the terms according to the equivalent SQL expression, and add them to the model. "Junk" terms that are neither *question* terms nor *project specific*, will be identified and removed.

3 Conclusion

The end result of this study is expected to be a greatly expanded list of synonyms and definitions for each of the SQL expressions in the domain model. This expanded set of domain terminology will enable TiQi to service future NL queries more accurately.

References

1. Mäder, P., Cleland-Huang, J.: A visual traceability modeling language. In: MoDELS (1). pp. 226–240 (2010)
2. Mäder, P., Cleland-Huang, J.: A visual language for modeling and executing traceability queries. Software and System Modeling 12(3), 537–553 (2013)
3. Mäder, P., Jones, P.L., Zhang, Y., Cleland-Huang, J.: Strategic traceability for safety-critical projects. IEEE Software 30(3), 58–66 (2013)
4. Pruski, P., Lohar, S., Aquanette, R., Ott, G., Amornborvornwong, S., Rasin, A., Cleland-Huang, J.: Tiqi: Towards natural language trace queries. In: IEEE 22nd International Requirements Engineering Conference, RE 2014, Karlskrona, Sweden, August 25-29, 2014. pp. 123–132 (2014), `http://dx.doi.org/10.1109/RE.2014.6912254`
5. Rempel, P., Mäder, P., Kuschke, T., Cleland-Huang, J.: Mind the gap: Assessing the conformance of software traceability to relevant guidelines. In: 36th International Conference on Software Engineering (ICSE) (2014)
6. Shin, Y., Cleland-Huang, J.: A comparative evaluation of two user feedback techniques for requirements trace retrieval. In: Proceedings of the ACM Symposium on Applied Computing, SAC 2012, Riva, Trento, Italy, March 26-30, 2012. pp. 1069–1074 (2012), `http://doi.acm.org/10.1145/2245276.2231943`